[illegible]

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/828,049	04/06/2001	Jason Souloglou		5766

36183 7590 02/13/2006

PAUL, HASTINGS, JANOFSKY & WALKER LLP
P.O. BOX 919092
SAN DIEGO, CA 92191-9092

EXAMINER
CHOW, CHIH CHING

ART UNIT	PAPER NUMBER
2192	

DATE MAILED: 02/13/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/828,049	SOULOGLOU ET AL.	
	Examiner	Art Unit	
	Chih-Ching Chow	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 09 November 2005.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-13,15,16,18 and 19 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-13,15,16,18 and 19 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 02 March 2005 is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date 8/24/01, 3/17/03.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to amendment dated November 09, 2005, per Applicant's request, claims 9, 18 and 19 are amended, and claims 14, 17, and 20 are canceled.

2. Claims 1-13, 15-16, 18-19 remain pending.

Double Patenting

3. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and, *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

4. Claims 1 and 10 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1 and 17 of copending Application No.09/827,971. Although the conflicting claims are not identical, they are not patentably distinct from each other, from the comparison listed in the following table:

Co-Application (09/827,971) US 2004/0205733A1	**Current Application (09/828,049) US 2002/0100030A1**

Claims	Claims
1. A method of generating an intermediate representation of program code, the method comprising the computer implemented steps of: (i)generating a plurality of register objects representing abstract registers, a single register object representing a respective abstract register; and (ii) generating expression objects each representing a different element of said program code as that element arises in the program code, each expression object being referenced by a register object to which it relates either directly, or indirectly via references from other expression objects.	1. A method for generating an intermediate representation of program code written for running on a programmable machine, said method comprising: (i) generating a plurality of register objects for holding variable values to be generated by the program code; and (ii) generating a plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code; wherein at least one variable sized register is represented by plural register objects, one register object being provided for each possible size of the variably sized register.
17. A system for generating an intermediate representation of program code, comprising: means for generating a plurality of register objects representing abstract registers, a single register object representing a respective abstract register; and means for generating expression objects each representing a different element of said program code as that element arises in the program code, each expression object being referenced by a register object to which it relates either directly, or indirectly via references from other expression objects.	10. A system for generating an intermediate representation of program code written for running on a programmable machine, the system comprising: means for generating a plurality of register objects for holding variable values to be generated by the program code; and means for generating a plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code; wherein at least one variably sized register is represented by plural register objects, one register object being provided for each possible size of the variably sized register.

5. Claim 1 of current application is anticipated by co-appliationclaim 1 in that co-application claim 1 contains all the limitations of the current application claim 1. Claim 1 of the current application therefore is not patentably distinct from co-application claim 1 and as such is unpatentable for obvious-type double patenting.

6. Claim 10 of current application is anticipated by co-appliationclaim 17 in that current application claim 10 contains all the limitations of the co-application claim 17. Claim 10 of the current application therefore is not patentably distinct from co-application claim 17 and as such is unpatentable for obvious-type double patenting.

7. This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Response to Arguments

8. Applicants' arguments for Claims 1-13, 15-16, 18-19 have been fully considered respectfully by the examiner but they are not persuasive.

9. Applicants' arguments are basically in the following points:

- **The Obviousness Rejection Over Aho In View of Davidson**

Examiner's Response: The argument about Aho's intermediate representation is an intermediate step between the source program and the garget program (REMARKS, pages 8-9), claim 1 is not concerned with the second stage, naming generating target code – this argument is valid, however, Davidson and Koizumi's teachings still read on the scope of current application. See 35 USC § 102 and 35 USC § 103 Rejections below.

- **Davidson Fails to Cure Aho's Deficiencies** – see REMARKS dated 11/09/05 page 11, first paragraph, "Clearly, the tuples of Davidson represent operations, variables and expressions derived from the source program. However, the tuples do not, in generating the intermediate representation, represent register objects as in claim 1".

Examiner's Response: Davidson's tuples are representation of operations, variables, and expressions, they are intermediate representations; they don't generate intermediate representations, the tuples use registers for their allocations, and they are addressable, see Davidson's column 3, lines 30-40, "The tuples are in ordered sequences within blocks, where a block is a part of the code that begins with a routine or label and ends in a branch, for example, where no entry or exit is permitted between the start and finish of a block. Each block is also a data structure, or node, and contains pointers to its successors and predecessors (these being to symbols in the symbol table). The interlinked blocks make up a flow graph, called the intermediate language graph, which is the representation of the program used by the back end to do the optimizations, register and memory allocations, etc"; and column 33, lines 20-25, "an operand field of a tuple node may contain the address of a symbol node, to represent the memory address (or the **register**) associated with that symbol." --Davidson's teaching, the front end part, already teaches the current application.

- **The Obviousness Rejeciton Over Aho in View of Koizumi, Claim 8**– see REMARKS dated 11/09/05 page 13, 6th paragraph, "Kiozumi does not disclose item (i) of claim 1".

Examiner's Response: claim 1 does not mention 'Abstract register' at all; Koizumi teaches claim 8 item (i), see 35 USC § 103 Rejections below.

Claim Rejections - 35 USC § 102

10. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

11. Claims 1, 2, 4-7, 10, 12-13, and 18-19 are rejected under 35 U.S.C. 102(b) as being anticipated by Davidson U.S. Pat. No. 5,613,117 by Davidson et al. (hereinafter "Davidson").

CLAIMS	**Davidson**
1. A method for generating an intermediate representation of program code written for running on programmable machine, said method comprising:	Davidson teaches an apparatus allows a plurality of register objects holding variable values. In Davidson, column 3, lines 14-20, "The front end scans and parses the source code modules (*program code*), and generates from them an intermediate language representation (*generating intermediate representation*) of the programs expressed in the source code. This intermediate language is constructed to represent any of the source code languages in a universal manner, so the interface between the front end and back end is of a standard format, and need not be rewritten for each language-specific front end."
(i) generating a plurality of register objects holding variable values be generated by the program code; and	For item (i), Davidson's column 2, lines 39-44, "Next in the internal organization of a compiler is the register and memory allocation. Up to this point, data references were to variables and constants by name or in the abstract, without regard to where stored; now, however, data references are assigned to more concrete locations, such as specific **registers** and memory displacements."
(ii) generating a plurality expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code;	For item (ii), see Davidson's column 33, lines 20-25, "an operand field of a tuple node may contain the address of a symbol node, to represent the **memory address** (or the **register**) associated with that symbol." And column 33, lines 41-47, "A data access tuple is a tuple which causes a **value** to be loaded from or stored into memory. (The word "memory" here includes **registers** in a register set of the target CPU 25. The only difference between a register and a normal memory location of the CPU 25 is that the 'address' of the register can only be used in a data access tuple.)"; Davidson further discloses this point, in Davidson,

	column 7, lines 43-49, “The expression may also be expressed as a **logic tree** as seen in FIG. 3, where the tuples are identified by the same reference numerals. This same line of source code could be expressed in assembly for a RISC type target machine, as three instructions LOAD, ADD integer, and STORE, using some register such as REG4 in the register file, in the general form seen in FIG. 3.” This paragraph shows generating the intermediate representation in terms of tuples that serve as expression objects. -- These quoted paragraphs taught us that the tuples serve as expression objects representing fixed values and the relationships between the fixed values.
(iii) wherein at least one variable sized register is represented by plural register objects, one register object being provided each possible size variably sized register.	For item (iii), Davidson discloses the concept of using variable sized register. In column 72, lines 22-25, “ALLOCATE.sub.--PERMANENT(operand, **size**) causes a permanent class TN (Temporary Name) of ‘size’ bytes to be created and referenced by the specified "operand" variable. If the "size" parameter is missing then the size of the TN is determined by the result data type of the current template.”
2. A method according to claim 1, wherein a write operation to a variably sized register is effected by writing to the register object corresponding to the appropriate size and maintaining a record of which register objects contain valid data.	For the feature of claims 1 see Claim 1 rejection. For the rest of claim 2 feature see Davidson, column 2, lines 45-52, “in the form of register allocation to maintain data in registers are minimize memory references; thus the program may be again rearranged to optimize register usage. Register allocation is also somewhat target machine dependent, and so the generic nature of the compiler must accommodate specifying the number, size and special assignments (*variably sized register*) for the register set of the target CPU.”

4. The method of claim 1, comprising translating the program code written for execution by a processor of a first type so that the program code may be executed by a processor of a second type, using the generated intermediate representation.	For the feature of claim 1 see rejection of claim 1. For the rest of claim 4 feature see Davidson, column 6, lines 57-62, "the front end 20 need not consider the architecture of the target machine 25 upon which the object code 23 will be executed, when the front end 20 is translating from **source code 15 to the internal representation** of interface 22, since the internal representation is independent of the target machine 25 architecture" – here the front end is a first type processor and the back end can be a second type of processor.
5. The method of claim 4, wherein the translation is performed dynamically as the program is run.	For the feature of claims 4 see claim 4 rejection, for the rest of claim 5 feature see Davidson column 2, lines 52-58, "Following register and memory allocation, the compiler implements the code generation phase, in which object code images are produced, and these are of course in the target machine language or instruction set, i.e., machine-specific. Subsequently, the object code images are linked to produce executable packages, adding various run-time modules".
6. The method of claim 1, comprising optimizing the program code by optimizing said generated intermediate representation.	For the feature of claim 1 see rejection of claim 1, for the rest of claim 6 see Davidson's column 3, lines 37-40, "The interlinked blocks make up a flow graph, called the intermediate language graph, which is the representation of the program used by the back end to do the optimizations".
7. The method of claim 6, wherein the optimizing step is used to optimize the program code written for execution by a processor of a first pipe so that the program	For the feature of claim 6, see rejection of claim 6. Since the intermediate representation, which was generated from the program code that was written for

code may be executed more efficiently by that processor.	execution by a processor (assuming it's first 'type' instead of 'pipe'), can be optimized, therefore the program code may be executed more efficiently by that processor.
10. A system for generating intermediate representation program code written for running on a programmable machine, the system comprising: (i) means for generating a plurality of register objects for holding variable values to be generated by the program code; and (ii) means for generating a plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code; (iii) wherein at least one variably sized register is represented by plural register objects, one register object being provided for each possible size of the variably sized register.	See rejection of claim 1.
12. (new) The method of claim 1, wherein said variably sized registers represented by separately addressable subsets of register objects.	For the feature of claim 1, see claim 1 rejection. For the rest of claim 12 feature see Davidson column 2, lines 48-59, "Register allocation is also somewhat target machine dependent, and so the generic nature of the compiler must accommodate specifying the number, size and special assignments for the register set of the target CPU. Following register and memory allocation, the compiler implements the code generation phase, in which object code images are produced, and these are of course in the target machine language or instruction set, i.e., machine-specific. Subsequently, the object code images are linked to produce executable packages,

	adding various run-time modules, etc., all of which is machine-specific"
13. (new) The method of claim 12, wherein said separately addressable subsets of register objects concurrently represent the same variably sized register.	Same as claim 12 rejection.
18. (new) The system of claim 10, wherein said variably sized register is represented by separately addressable subsets of register objects.	For the feature of claim 10, see rejection of claim 10. For the rest of claim 18 feature see claim 12 rejection.
19. (new) The sytem of claim 18, wherein said separately addressable subsets of register objects concurrently represent the same variably sized register.	For the feature of claim 18, see rejection of claim 18. For the rest of claim 19 feature see claim 12 rejection.

Claim Rejections - 35 USC § 103

12. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

13. Claims 3 is rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,613,117 by Davidson et al. (hereinafter "Davidson"), in view of Aho et al, "Compiler, principles, techniques, and tools" book, published in 1986 (herein after "Aho").

CLAIMS	**Davidson / Aho**
3. A method according to claim 2, wherein a read operation from a variably sized register is effected by determining from said record if there is valid data in more than one corresponding register object which must be combined to give the same effect as reading from the variable register, and (i) if it is determined that such combination is required, reading from the appropriate register object; and (ii) if it is determined that such combination is required, combining the contents appropriate register objects to provide a read value.	For the feature of claim 2 see claim 2 rejection. Davidson teaches all aspects of the applicant's claims but it does not specifically mention 'combining variable sized variables". However, Aho teaches it in an analogous prior art. For item (i) and (ii), Aho discloses the skill to use multiregister for operations. See Aho, page 565, 'Multiregister Operations', "We can modify our labeling algorithm to handle operations like multiplication, division, or a function call, which normally require more than one register to perform. Simply modify step (6) of Fig. 9.23, the labeling algorithm, so label (n) is always at least the number of the registers required by the operation." The function 'label' is able to return the number of the registers required by the operation. In addition to the 'Multiregister Operation', Aho also mentioned 'register pair' concept, page 517, under 'Register Allocation', 5th paragraph, "Certain machines require register-pairs (an even and next odd-numbered register) for some operands and results." Basically, registers can be defined in 8, 16, 32, 64 bits, it can always come in multiples. Therefore it's able to cover the function of combining many appropriate register objects needed for a certain read value. It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the **Multiregister operation** of Aho with the reading and writing operations further taught by Davidson, for the purpose of allowing value to be **loaded from** or **stored into** multiple registers. See Aho, page 559, Fig. 9.20, each of the t2, t3, t1 and t4 are 'expression objects' and they

are feed into more than one register object. (E.g. t1 is an expression object, it feed into register objects a and b; t2, is also an expression object, it feed into register objects c and d). Any program would have some expression objects since the program should perform certain functions, functions are 'operations' and they are represented by 'expression objects'; operands feed into operations, here operands are represented by 'register objects'.
It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the **intermediate representation** of Davidson with combining multiple registers taught by Aho, for the purpose of performing computation for a program (See Aho, page 559, 2nd paragraph).

14. Claims 8-9, 11, 15-16 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Pat. No. 5,613,117 by Davidson et al. (hereinafter "Davidson"), in view of in view of Aho et al, "Compiler, principles, techniques, and tools" book, published in 1986 (herein after "Aho"), further in view of U.S. Patent No. 5,586,323 by Shinobu Koizumi et al. (hereinafter "Koizumi").

CLAIM	**Davidson / Aho / Koizumi**
8. A method of generating an intermediate representation of program code expressed in terms of the instruction set of a subject processor comprising at least one variable sized register, the method comprising the computer implemented steps (i) generating a set of associated abstract register objects representing the variable sized register; (ii) for each write operation of a certain field width to the variable sized register, writing an abstract register of the same width; (iii) maintaining a record of which abstract register objects contain valid data, which record is updated upon each write operation; and (iv) for each read operation a given field width, determining from said record whether there is valid data in more than one of said different sized abstract registers of the set which must be combined to give the same effect as the same read operation performed upon the variable size register; and (a) if it is determined that no combination is so required, reading directly from the appropriate register; or (b) if it is determined that data from more than one register must be so combined, combining the contents of those registers.	For items (ii), (iii), and (iv) see claim 1 rejection. Davidson teaches all aspects of claim 8 but it does not specifically mention item (i) 'abstract register'. However, Kiozumi teaches 'abstract register' in an analogous prior art. In Koizumi column 4, lines 53-58, "an **abstract register** machine (also referred to as ARM or Arm in abbreviation) having a plurality of registers is presumed, wherein an instruction sequence for the **abstract register** machine or ARM is made use of as a basic part of the common object program (referred to as the abstract object program)". For items (a) and (b) see claim 3 rejection. It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the **intermediate representation** of Davidson, and combining registers taught by Aho, with the abstract register taught by Kiozumi, for the purpose of preserving a form of a program to be executed repeatedly (See Koizumi, column 2, lines 66-67).
9. The method according to claim 8, wherein the step of determining whether or not the contents of more than one abstract register must be combined and if so which abstract registers must be combined, is determined in accordance with following conditions respect each set of different sized abstract registers:	For the feature of claim 8, see rejection of claim 8. For the rest of the claim, see the rejection of claims 1, 3, and 8. With respect to item (i) and (ii) see rejection of claim 1 (i) and (ii), where recited that "A data access tuple is a tuple which causes a **value** to be loaded from or stored into memory. (The word "memory"

(i) if the data required for an access lies wholly within one valid abstract register, that register only is accessed; and (ii) if the data required for an access lies within more than one valid abstract register, data is combined from those valid abstract registers to perform the access.	here includes **regist)**". Only the required size is allocated, if data lies wholly within one valid abstract register, that register only is accessed. If the data required for an access lies within more than one register, that much amount of register space would be allocated (i.e. multiple registers may be combined).
11. A system for generating an intermediate representation of program code expressed in terms of the instruction set of a subject processor comprising of at least one variably sized register, the system comprising: (i) means for generating set of associated abstract register objects representing the variably sized register; (ii) means for writing, for each write operation of a certain field width to the variable sized register to an abstract register object of the same width; (iii) means for maintaining a record of which abstract register objects contain valid data, the record being updated upon each write operation; and (vi) means for determining from said record, for each read operation of a given width, whether there is valid data in more than one said different sized abstract registers of the set which must be combined to give the same effect as the same read operation performed upon the variable size register, and (a) if is determined that no combination is so required, reading directly from the appropriate register; or (b) if it is determined that data from more	See rejection of claim 8. For item (a) and (b) see rejections of claim 3 (i) Multiregister Operation, and claim 9 (i) and (ii).

than one register must be so combined, combining the contents of those registers.	
15. (new) The method of claim 8, wherein said variably sized register is represented by separately addressable subsets of abstract register objects.	For the feature of claim 8, see rejection of claim 8. For the rest of claim 15 feature see claim 12 rejection.
16. (new) The method of claim 15, wherein said separately addressable subsets of abstract register objects concurrently represent the same variably sized register.	For the feature of claim 15, see rejection of claim 15. For the rest of claim 16 feature see claim 12 rejection.

Conclusion

The following summarizes the status of the claims:

35 USC § 102 rejection: Claims 1, 2, 4-7, 10, 12-13, and 18-19

35 USC § 103 rejection: Claims 3, 8, 9, 11, 15, 16

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 7:30am - 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300. Any inquiry of a general nature of relating to the status of this application should be directed to the **TC2100 Group receptionist: 571-272-2100**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chih-Ching Chow
Examiner
Art Unit 2192
February 03, 2005

CC

[illegible]

TUAN DAM
SUPERVISORY PATENT EXAMINER